4-8-2021

Lecture 08: Partial Eigen Decomposition of Large Symmetric Matrices via Thick-Restart Lanczos with Explicit External Deflation and its Communication-Avoiding Variant

Zhaojun Bai
University of California, Davis

Partial Eigen Decomposition of Large Symmetric Matrices via Thick-Restart Lanczos with Explicit External Deflation and its Communication-Avoiding Variant

Zhaojun Bai
University of California, Davis

46th Annual Spring Lecture Series, Univ. of Arkansas
April 5-9, 2021 (Virtual)

Collaborators

- Jack J. Dongarra, Univ of Tennessee
- Chao-Ping Lin, UC Davis
- Ding Lu, Univ of Kentucky
- Ichitaro Yamazaki, Sandia National Labs.

Introduction

1. Problem statement ("many eigenpairs computation"):

   Let (λ_i, v_i) *be the eigenpairs of a* $n \times n$ *symmetric matrix* A *with the ordering* $\lambda_1 \leq \lambda_2 \leq \ldots$. *Compute the partial eigen decomposition:*

   $$AV_m = V_m \Lambda_m,$$

   where $V_m = [v_1, v_2, \ldots, v_m]$ *and* $\Lambda_m = diag(\lambda_1, \lambda_2, \ldots, \lambda_m)$, n *is huge and* m *is large.*

2. Emerging applications such as
   - electronic structure calculations of Lithium-ion electrolyte,
   - graphene,
   - dyanmics analysis of viral capsids of supramolecular systems such as Zika and West Nile viruses
   - ...

Introduction

3. Approaches – *an incomplete list*:

   - Full eigenvalue decomposition:
     - LAPACK, ScaLAPACK, PLASMA, MAGMA
     - ELPA (Eigenvalue Solves for Petaflop Applications) https://elpa.mpcdf.mpg.de/
     - EigenExa, https://www.r-ccs.riken.jp/labs/lpnctrt/projects/eigenexa/
     - ...
     - QDWHeig, [Sukkari, Ltaief, Keyes at KAUST]
     - SLATE, [Gates *et al*, U of Tennessee]

     Stable, but expensive, $O(n^2)$ storage and $O(n^3)$ flops

   - "Spectrum slicing:"
     - SLEPc, https://slepc.upv.es/
     - EVSL, http://www.cs.umn.edu/∼saad/software
     - FEAST, http://www.ecs.umass.edu/∼polizzi/feast/
     - z-Pares, http://zpares.cs.tsukuba.ac.jp/
     - ...
     - SISLICE: [Williams-Young, Yang, LBNL]

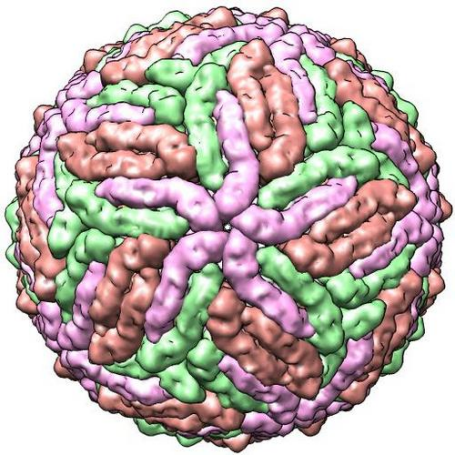
     Scalable, but several parameters (e.g. windows) and duplicate/missing eigenvalues on the interface.

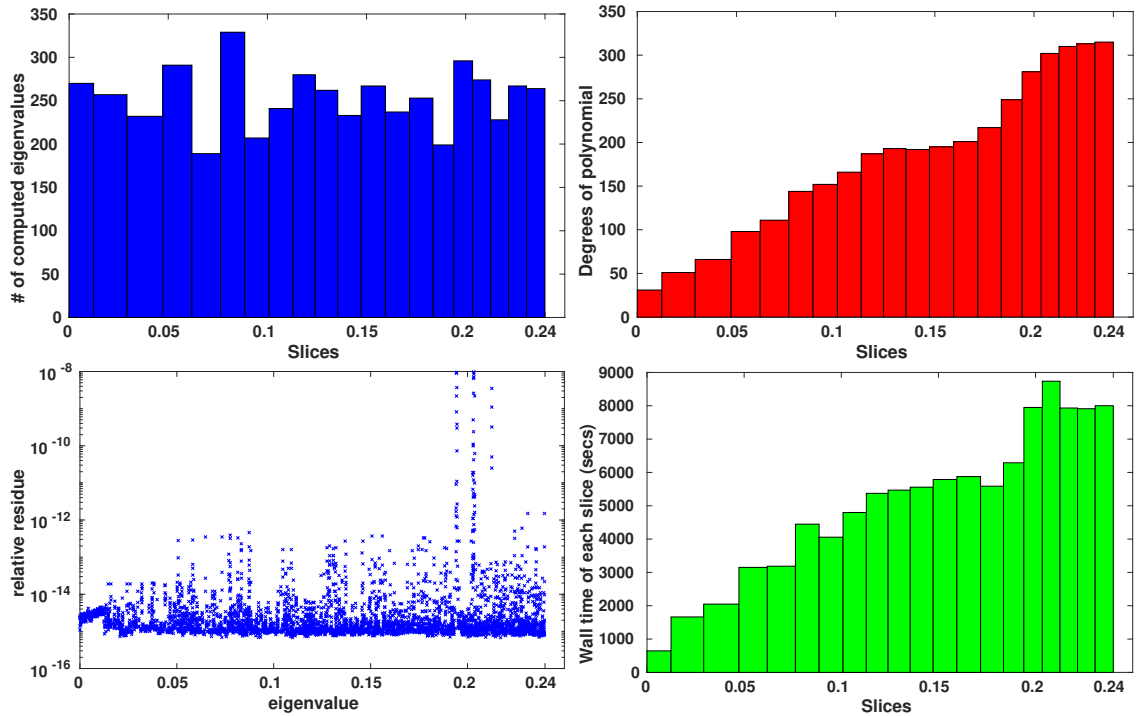
Introduction

4. Using Lanczos or any other subspace projection methods for many eigenpairs are challenging – numerically and computationally:
   - require (internal) locking (deflation) to avoid computing the same eigenvalues (nm^2 flops)
   - often needs large subspace (memory) (e.g., $m = 2n_e$)

Introduction

5. Challenging problem – a case demo
   preliminary results with **EVSL** (http://www.cs.umn.edu/∼saad/software)





Left: Dengue virus `https://www.rcsb.org/structure/4cct`. The matrix dimension is 307,260 in an elastic network model.
Middle: DOS sliced $5,076$ eigenvalues into 20 subintervals (top) and the relative residual norms of eigenpairs (bottom). $\|\widehat{V}^T\widehat{V} - I\|_F = O(10^{-6})$.
Right: Degrees of filter polynomials (top) CPU timing for each slices

Introduction

6. Many eigenpairs computation is challenging even when vast computational resources are available.
7. This talk is about an on-going development of

   Lanczos + Explicit external deflation + Communication-avoiding

   for computing many eigenpairs.

Rest of the talk

I. Explicit External Deflation (EED)
II. Backward stability of EED
III. Communication-avoiding algorithm for MPK (CA-MPK)
IV. Eigensolver (TRLan) with EED and CA-MPK
V. Concluding remarks
VI. References

I. Explicit External Deflation (EED)

I. Explicit External Deflation (EED)

1. EED [Hotelling'33, Wielandt'44]

Let $AV = V\Lambda$ be the eigen-decomposition of A, partition

$$V = [V_k,\, V_{n-k}] \quad \textit{and} \quad \Lambda = \begin{bmatrix} \Lambda_k & \\ & \Lambda_{n-k} \end{bmatrix},$$

where V_k consists of eigenvectors corresponding to first k eigenvalues. Define

$$\widehat{A} = A + \sigma V_k V_k^T$$

Then
(a) The eigenvalues of $\widehat{A}$ are

$$\lambda_i(\widehat{A}) = \begin{cases} \lambda_i + \sigma & \textit{for } 1 \le i \le k \\ \lambda_i & \textit{for } k+1 \le i \le n \end{cases}$$

(b) A and $\widehat{A}$ have the same eigenvectors

Therefore, after the first k eigenvalues are computed, one can pick a proper shift σ to move these eigenvalues away, and then compute the next k eigenvalues of A using the matrix $\widehat{A}$.

I. Explicit External Deflation (EED)

2. Governing equations of j steps of exact EED process: for $j = 1, 2, \ldots,$

$$A_j = A_{j-1} + \sigma_j v_j v_j^T = A + V_j \Sigma_j V_j^T \quad \text{and } A_0 = A.$$
$$A_j v_{j+1} = \lambda_{j+1} v_{j+1}$$
$$A_j V_j = V_j (\Lambda_j + \Sigma_j)$$

   with initial $Av_1 = \lambda_1 v_1$, where $\Sigma_j = \text{diag}(\sigma_1, \ldots, \sigma_j)$.

3. From the governing equations, we have the desired partial eigen decomposition

$$AV_j = V_j \Lambda_j.$$

I. Explicit External Deflation (EED)

4. Advantages of EED:

- No need to explicitly orthogonalize the projection subspace to the computed eigenvectors in contrast to the implicit internal deflation ("locking")
- Easily incooperated into existing eigensolvers, such as TRLan and ARPACK.
- Straightforward extension to generalized symmetric eigenproblem $Av = \lambda Bv$:

$$(A + \sigma B V_k V_k^T B)x = \lambda Bx$$

  without factoring/inverting the matrix B.
- Readily exploit structures of A and B, for example, see EED for the linear response eigenvalue problem [Bai-Li-Lin'17].
- Extensible to other eigenvalue-type problems, such as SVD.
- Nonlinear eigenvector problem (NEPv) analogy: level shifting

I. Explicit External Deflation (EED)

5. **Two key issues of EED:**
   (a) Numerical stability with approximate eigenvectors $\widehat{V}_k$:

$$\widehat{A}_j = A + \widehat{V}_j \Sigma_j \widehat{V}_j^T$$

   (b) Cost of matrix powers kernel (MPK) for generating Krylov subspace:

$$\left[p_0(\widehat{A}_j)v_0,\, p_1(\widehat{A}_j)v_0,\, \ldots,\, p_s(\widehat{A}_j)v_0\right]$$

   where $\{p_k(\cdot)\}$ are recursively defined polynomials.

II. Backward stability of EED

II. Backward stability of EED

1. Previous work: [Wilkinson'65], [Parlett'82], [Saad'89], [Jang/Lee'06]
2. Governing equations of j steps *inexact* EED:

$$\begin{aligned}
\widehat{A}_j &= \widehat{A}_{j-1} + \sigma_j \widehat{v}_j \widehat{v}_j^T = A + \widehat{V}_j \Sigma_j \widehat{V}_j^T \\
\widehat{A}_j \widehat{v}_{j+1} &= \widehat{\lambda}_{j+1} \widehat{v}_{j+1} + \eta_{j+1} \\
\widehat{A}_j \widehat{V}_j &= \widehat{V}_j (\widehat{\Lambda}_j + \Sigma_j) + \widehat{V}_j \Sigma_j \Phi_j + E_j
\end{aligned}$$

where $\widehat{A}_0 = A$,

$$\begin{aligned}
\|\eta_{j+1}\| &\leq tol \cdot \|A\| \\
\widehat{\Lambda}_j &= \mathsf{diag}(\widehat{\lambda}_1, \ldots, \widehat{\lambda}_j) \\
\widehat{V}_j &= [\widehat{v}_1, \ldots, \widehat{v}_j], \\
\Sigma_j &= \mathsf{diag}(\sigma_1, \ldots, \sigma_j), \\
\Phi_j &= \mathsf{utri}(\widehat{V}_j^T \widehat{V}_j - I_j), \\
E_j &= [\eta_1, \eta_2, \ldots, \eta_j].
\end{aligned}$$

and tol is prescribed relative residual tolerance.

II. Backward stability of EED

3. Notion of backward stability for computed eigenpairs $(\widehat{\Lambda}_{j+1}, \widehat{V}_{j+1})$ with relative residual tolerance tol:

   - The loss of orthogonality

   $$\omega_{j+1} = \|\widehat{V}_{j+1}^T \widehat{V}_{j+1} - I\|_F = O(tol)$$

   - The backward error

   $$\delta_{j+1} = \min_{\Delta \in \mathcal{H}} \|\Delta\|_F = O(tol \cdot \|A\|),$$

   where

   $$\mathcal{H} = \left\{ \Delta \mid (A+\Delta)Q_{j+1} = Q_{j+1}\widehat{\Lambda}_{j+1},\ \Delta = \Delta^T,\ Q_{j+1} = \mathsf{orth}(\widehat{V}_{j+1}) \right\}$$

II. Backward stability of EED

4. Two key quantities
   - Spectral gap
   $$\gamma_j \equiv \min_{\lambda \in \mathcal{I}_{j+1}, \theta \in \mathcal{J}_j} |\lambda - \theta|,$$
   where $\mathcal{I}_{j+1} = \{\widehat{\lambda}_1, \ldots, \widehat{\lambda}_j, \widehat{\lambda}_{j+1}\}$ is the set of computed eigenvalues, and $\mathcal{J}_j = \{\widehat{\lambda}_1 + \sigma_1, \ldots, \widehat{\lambda}_j + \sigma_j\}$ is the set of computed eigenvalues with shifts.
   - Shift-gap ratio
   $$\tau_j \equiv \frac{1}{\gamma_j} \cdot \max_{1 \le i \le j} |\sigma_i|.$$

5. Theorem (Rule of Thumb)

   Under mild assumptions, if $\gamma_j^{-1}\|A\| = O(1)$ *and* $\tau_j = O(1)$, *then*

   $$\omega_{j+1} = O(tol) \quad \textit{and} \quad \delta_{j+1} = O(tol \cdot \|A\|).$$

6. Dynamical choice of shifts σ_j to satisfy the conditions $\gamma_j^{-1}\|A\| = O(1)$ and $\tau_j = O(1)$.

II. Backward stability of EED

7. Numerics of TRLED = TRLan + EED:

- Test matrices:

matrix	n	$[\lambda_{\min}, \lambda_{\max}]$	$[\lambda_{\text{low}}, \lambda_{\text{upper}}]$	n_e
Laplacian	$40,000$	$[0, 7.9995]$	$[0, 0.07]$	205
worms20	$20,055$	$[0, 6.0450]$	$[0, 0.05]$	289
SiO	$33,401$	$[-1.6745, 84.3139]$	$[-1.7, 2.0]$	182
Si34H36	$97,569$	$[-1.1586, 42.9396]$	$[-1.2, 0.4]$	310
Ge87H76	$112,985$	$[-1.214, 32.764]$	$[-1.3, -0.0053]$	318
Ge99H100	$112,985$	$[-1.226, 32.703]$	$[-1.3, -0.0096]$	372

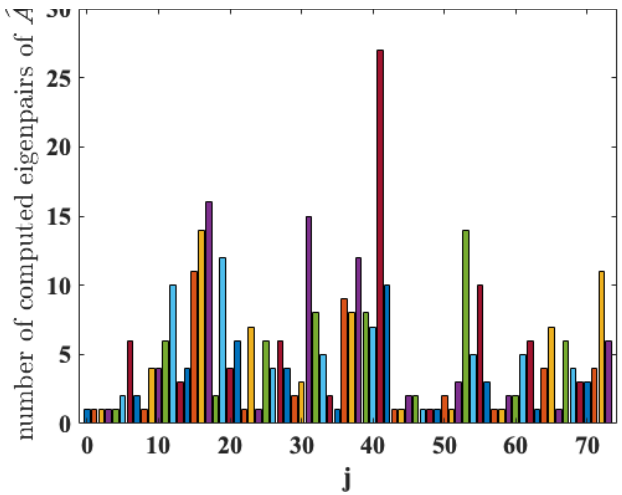
II. Backward stability of EED

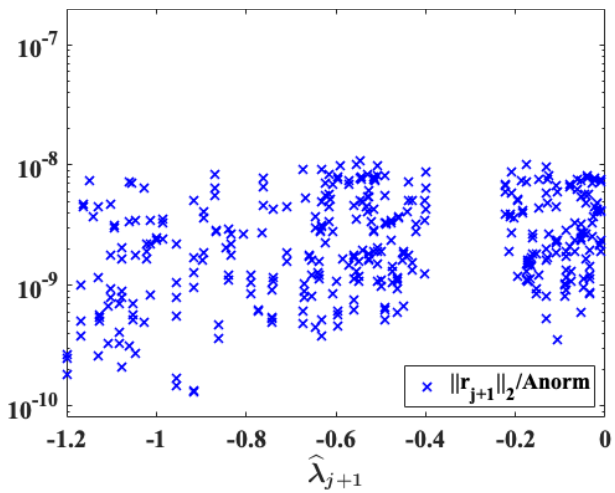
7. Numerics of TRLED = TRLan + EED, *cont'd*

- Results:

matrix	$\widehat{n}_e$	$j_{\max}$	$\omega_{\widehat{n}_e}$	$\|R_{\widehat{n}_e}\|_{\mathrm{F}}/\texttt{Anorm}$	CPU time (sec.)	
					TRLED	TRLan
`Laplacian`	205	60	$1.93 \cdot 10^{-8}$	$6.33 \cdot 10^{-8}$	66.5	86.0
`worms20`	289	86	$2.63 \cdot 10^{-8}$	$7.24 \cdot 10^{-8}$	57.3	74.8
`SiO`	182	41	$2.33 \cdot 10^{-8}$	$4.71 \cdot 10^{-8}$	42.4	47.1
`Si34H36`	310	72	$3.41 \cdot 10^{-8}$	$7.50 \cdot 10^{-8}$	309.9	310.4
`Ge87H76`	318	66	$4.08 \cdot 10^{-8}$	$8.50 \cdot 10^{-8}$	388.7	421.0
`Ge99H100`	372	74	$3.65 \cdot 10^{-8}$	$7.63 \cdot 10^{-8}$	501.1	533.4

- EED profile of `Ge99H100`:





II. Backward stability of EED

7. Numerics of TRLED = TRLan + EED, *cont'd*

- Observations:

  (1) All desired eigenvalues are successfully computed: $n_e = \widehat{n}_e$

  (2) computed eigenpairs are backward stable:
  $$\omega_{n_e} = O(tol) \text{ and } \delta_{n_e} \approx \|R_{n_e}\| = O(tol \cdot \|A\|).$$

  (3) EED steps do not significant slow down in execution time, in fact, slightly improved due to low "internal" memory usage.

III. Communication-avoiding algorithm for computing MPK

III. Communication-avoiding algorithm for computing MPK

1. Sparse-plus-low-rank matrix powers kernel (MPK)

$$[p_0(B)v, p_1(B)v, \ldots, p_s(B)v]$$

with

$$B = A + \sigma U_k U_k^T = \text{sparse} + \text{low rank}$$

and $AU_k = U_k \Lambda_k$, $U_k^T U_k = I_k$ and $p_j(\cdot)$ are polynomials defined *recursively*.

2. For simplicity, consider polynomials $\{p_j(\cdot)\}$ in the monomial basis and compute the MPK:

$$\begin{aligned}[p_0(B)v, p_1(B)v, \ldots, p_s(B)v] &= \left[x, Bx, B^2x, \ldots, B^s x\right] \\ &\equiv \left[x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(s)}\right]\end{aligned}$$

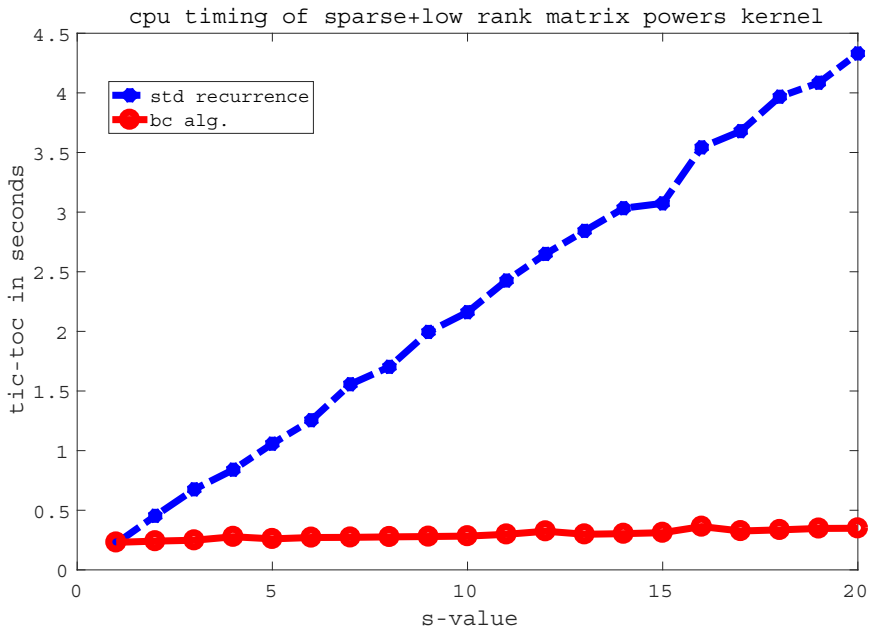
3. Standard MPK algorithm

1: $x^{(0)} = x$;
2: **for** $j = 1 : s$ **do**
3: $\quad x^{(j)} = Ax^{(j-1)} + \sigma(U_k(\underline{U_k^T x^{(j-1)}}))$
4: **end for**

III. Communication-avoiding algorithm for computing MPK

4. Example
   - $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
   - matrix powers kernel $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
   - Timing of the standard algorithm in MATLAB – blue curve.

$n = 250\,000$ and $k = 1\,000$



   - **Q: what's the red curve?**

III. Communication-avoiding algorithm for computing MPK

5. An algorithm has two costs:

   arithmetic (flops) + movement of data (communication)

6. Communication is the bottleneck on modern architectures.
7. Q: How to exploit the sparse-plus-low-rank matrix structure to reduce communication cost?

   Ans.: *use a specialized communication-avoiding algorithm developed by Leiserson-Rao-Toledo'97 (*"out-of-core"*) and Knight-Carson-Demmel'13 (*"exploiting data sparsity"*)*

III. Communication-avoiding algorithm for computing MPK

8. **Communication-Avoiding (CA) algorithm** for the MPK

1: $x^{(0)} = x$
2: $b_0 = U_k^T x^{(0)}$
3: $W_k^{(j)} = \Lambda_k^j + \sigma \sum_{i=1}^{j} \Lambda_k^{i-1}(\Lambda_k + \sigma)^{j-i}$ *for* $j = 1 : s-1$,
4: $b_j = W_k^{(j)} b_0$ *for* $j = 1 : s-1$
5: $[c_0, c_1, \ldots, c_{s-1}] = U_k[b_0, b_1, \ldots, b_{s-1}]$
6: **for** $j = 1 : s$ **do**
7: $\quad x^{(j)} = Ax^{(j-1)} + \sigma c_{j-1}$
8: **end for**

9. Benefits of CA-MPK algorithm:
   - Reduced flops

$$\text{flops}_{\text{std}} = nnzA \cdot s + nks + nks$$
$$\text{flops}_{\text{ca}} = nnzA \cdot s + nks + nk + O(k^2 s)$$

   - Reduced movement of data: U_k is only accessed twice.

III. Communication-avoiding algorithm for computing MPK

10. Derivation of a simplified blocking covers (communication-avoiding) MPK algorithm

- Recall the identity $C^j = (E+F)^j = E^j + \sum_{i=1}^{j} E^{i-1} F C^{j-i}$
- Let $B = A + \sigma U_k U_k^T$ and $AU_k = U_k \Sigma_k$, then for $j = 1, \ldots, s$,

$$B^j = \left(A + \sigma U_k U_k^T\right)^j = A^j + \sigma \sum_{i=1}^{j} \underline{A^{i-1} U_k} U_k^T B^{j-i} = A^j + \sigma \sum_{i=1}^{j} \underline{U_k \Lambda_k^{i-1}} U_k^T B^{j-i}$$

- Then for $j = 0, 1, \ldots, s-1$,

$$U_k^T B^j x = \underline{U_k^T A^j} x + \sigma \sum_{i=1}^{j} \Lambda_k^{i-1} \underline{U_k^T B^{j-i}} x = \underline{\Lambda_k^j U_k^T} x + \sigma \sum_{i=1}^{j} \Lambda_k^{i-1} \underline{(\Lambda_k + \sigma)^{j-i} U_k^T} x$$

$$= \left[\Lambda_k^j + \sigma \sum_{i=1}^{j} \Lambda_k^{i-1} (\Lambda_k + \sigma)^{j-i}\right] \underline{U_k^T x} \equiv W_k^{(j)} \cdot \underline{U_k^T x} \equiv b_j$$

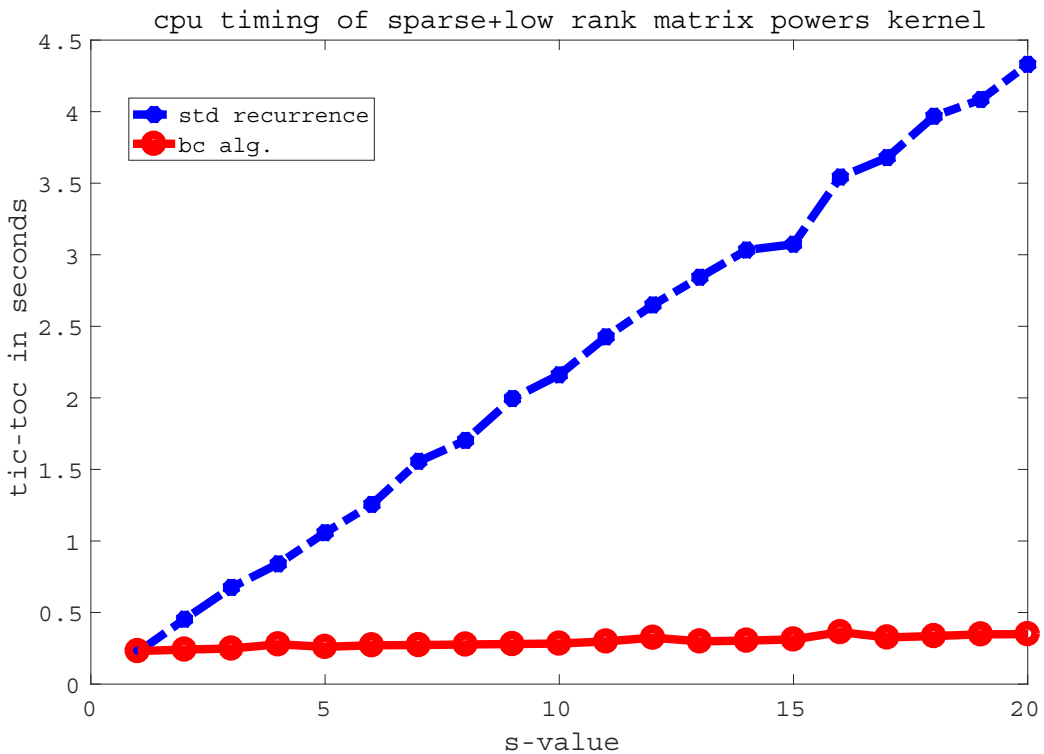
- Compute $[c_0, c_1, \ldots, c_{s-1}] = U_k [b_0, b_1, \ldots, b_{s-1}]$
- Finally for $j = 1, \ldots, s$,

$$\begin{aligned} x^{(j)} &\equiv B^j x = B(B^{j-1} x) = (A + \sigma U_k U_k^T)(B^{j-1} x) \\ &= A(B^{j-1} x) + \sigma U_k U_k^T (B^{j-1} x) = A(B^{j-1} x) + \sigma U_k \left(U_k^T B^{j-1} x\right) \\ &= A x^{(j-1)} + \sigma \underline{U_k b_{j-1}} = A x^{(j-1)} + \sigma c_{j-1} \end{aligned}$$

III. Communication-avoiding algorithm for computing MPK

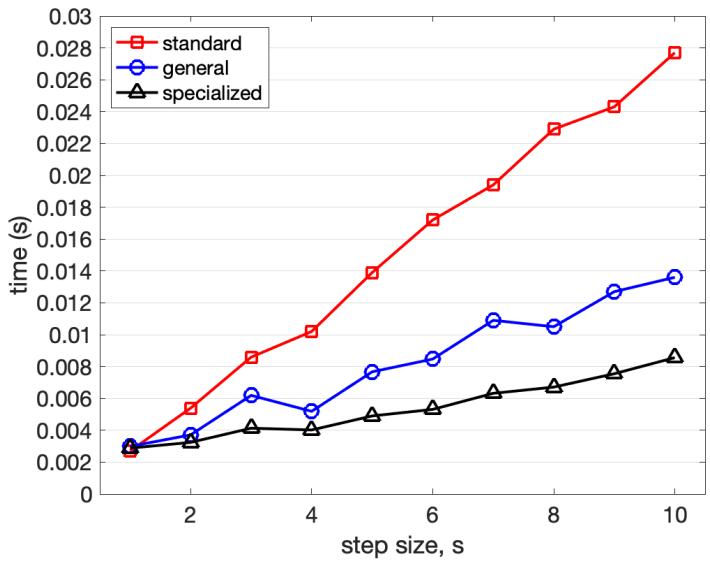
11. Example, *cont'd*:
    - $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
    - Matrix powers kernel $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
    - Timing in MATLAB
        - Standard algorithm – blue curve
        - CA algorithm – red curve

$n = 250000$ and $k = 1000$

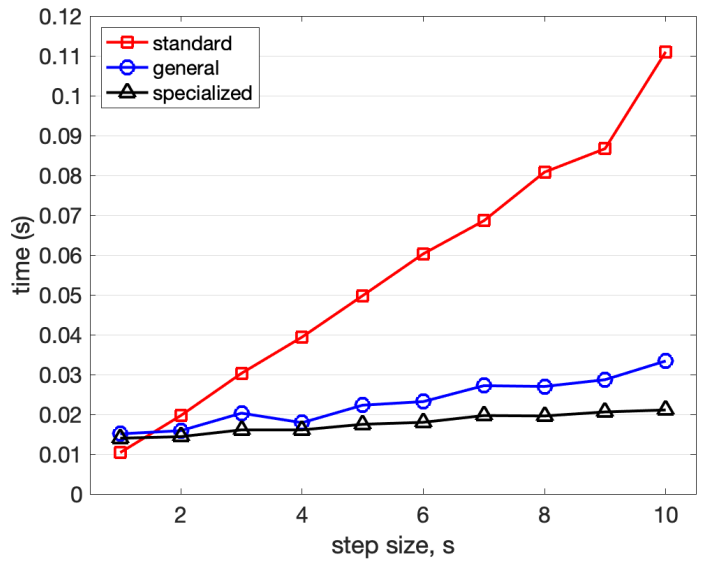


III. Communication-avoiding algorithm for computing MPK

11. Example, *cont'd*:
    - $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
    - Matrix powers kernel $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
    - Timing in one node (32 cores) of Cori (NERSC)



$k = 100$



$k = 200$

12. Rounding error analysis of CA-MPK – *skip*.

IV. Lanczos algorithm with EED and MPK

IV. Lanczos algorithm with EED and MPK

1. Lanczos algorithm
   - Lanczos process
     $$AQ_j = Q_j T_j + \beta_j q_{j+1} e_j^T$$
     where span$\{Q_j\}$ = span$\{q, Aq, \ldots, A^{j-1}q\}$ (Krylov subspace)
   - Rayleigh-Ritz approximation
     $$T_j x_i = \theta_i x_i$$
     $$(\lambda_i, v_i) \approx (\theta_i, Q_j x_i)$$
2. Lacnzos algorithm is efficient for computing a few exterior eigenvalues (and eigenvectors).
3. Two main kernels
   - Matrix-Vector multiply (SpMV) for generating Krylov subspace: Aq
   - Re-orthogonalization for maintaining orthonormal basis vectors: Q_j
4. Two variants of Lanczos method:
   - Thick-restart Lanczos (TRLan) – to reduce memory requirements
   - s-step Lanczos (s-Lanczos) – replace SpMV by MPK (Matrix Powers Kernel) to reduce communication cost

IV. Lanczos algorithm with EED and MPK

5. sTRLED = s-step-TRLan + EED + CA-MPK

set $\mathbf{q}_1 = \mathbf{q}/\|\mathbf{q}\|_2$, $k = 0$.
for $j = 1, 2, 3, \ldots$

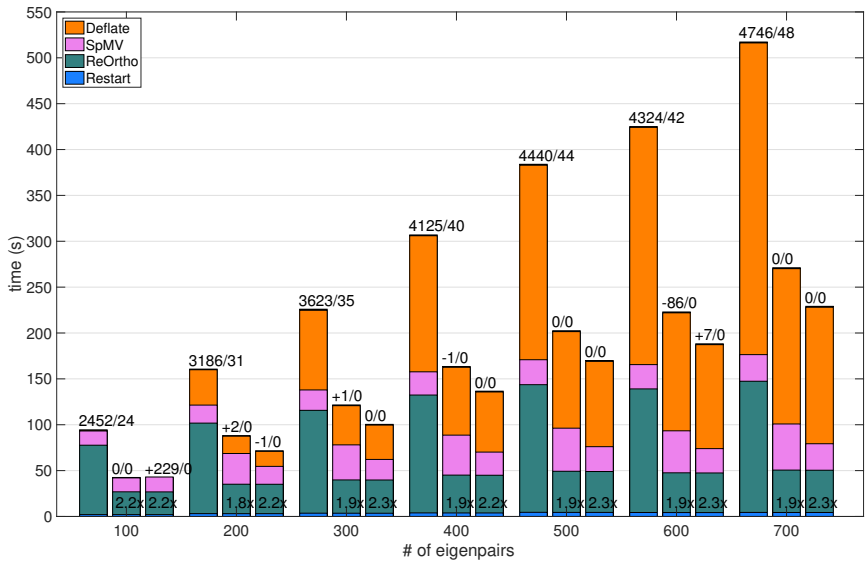
1. Initialization.
   a. $\mathbf{p} := (A + \alpha U_d U_d^H)\mathbf{q}_{k+1}$
   b. $\alpha_{k+1} := \mathbf{q}_{k+1}^H \mathbf{p}$
   c. $\mathbf{p} := \mathbf{p} - \alpha_{k+1}\mathbf{q}_{k+1} - \sum_{i=1}^{k} \beta_i \mathbf{q}_i$
   d. $\beta_{k+1} := \|\mathbf{p}\|_2$
   e. $\mathbf{q}_{k+2} := \mathbf{p}/\beta_{k+1}$
2. The j-th restart-loop.
   for $i = k + 2 : s : m$
   a. Starting vector $\mathbf{p}_i = \mathbf{q}_i$.
   b. Matrix Powers Kernel:
      for $\ell = i, i + 1, \ldots, i + s - 1$
         $\mathbf{p}_{\ell+1} := (A + \alpha U_d U_d^H)\mathbf{p}_\ell$
      end for
   c. Block three-term orthogonalization:
      $R_{i-s:i,\ i+1:i+s} := Q_{i-s:i}^H P_{i+1:i+s}$
      $P_{i+1:i+s} := P_{i+1:i+s} - Q_{i-s:i} R_{i-s:i,\ i+1:i+s}$
   d. Tall-skinny Cholesky QR factorization:
      $B := P_{i+1:i+s}^H P_{i+1:i+s}$
      $R_{i+1:i+s,\ i+1:i+s} := \mathrm{chol}(B)$
      $Q_{i+1:i+s} := P_{i+1:i+s} R_{i+1:i+s,i+1:i+s}^{-1}$
   e. Reorthogonalize $Q_{i+1:i+s}$ if necessary:
      Classical Gram Schmidt followed by Cholesky QR.
   f. Update the projected matrix T_m:
      see, e.g., [4, Sec. 4.2.2].
   end for
3. The j-th restart.
   a. compute all eigenpairs of T_m and the corresponding residual norms for Ritz pairs by (2).
   b. if stopping criteria is satisfied then
   c. compute desired Ritz vectors and exit.
   d. else restart:
   e. update k (see [14], [13]).
   f. select k Ritz values $\lambda_1, \ldots, \lambda_k$ of interest, and compute their Ritz vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$.
   g. set $\alpha_i = \lambda_i$ and $\beta_i = \|A\mathbf{q}_i - \lambda_i \mathbf{q}_i\|_2$ by (2), for $i = 1, \ldots, k$,
   h. set $\mathbf{q}_{k+1} = \mathbf{q}_{m+1}$.
   i. end if

end for

IV. Lanczos algorithm with EED and MPK

6. Preliminary results on performance of
   sTRLED $=$ s-step-TRLan $+$ EED $+$ CA-MPK

   Speedup on single core for `Si34H26` with Standard, Block-cover and Specialized MPKs:

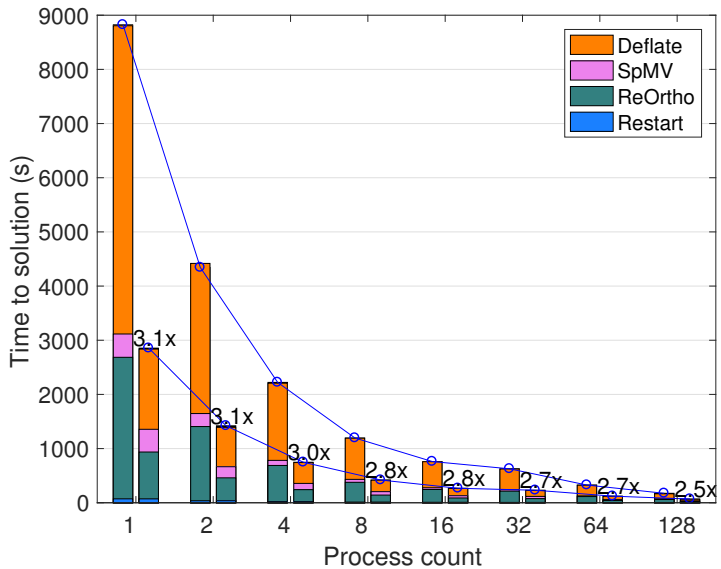


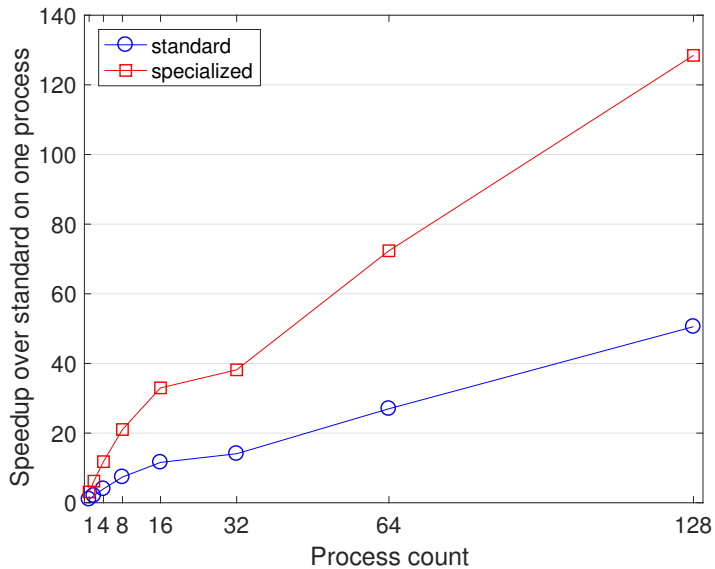
$n = 97569$, $n_e = 100, \ldots, 700$
computing 100 eigenvalues at a time with $m = 200$ and $s = 5$

IV. Lanczos algorithm with EED and MPK

7. Preliminary results on strong-parallel scaling of
   sTRLED $=$ s-step-TRLan $+$ EED $+$ CA-MPK

   speedup over standard MPK on one processor for `Si87H76`:



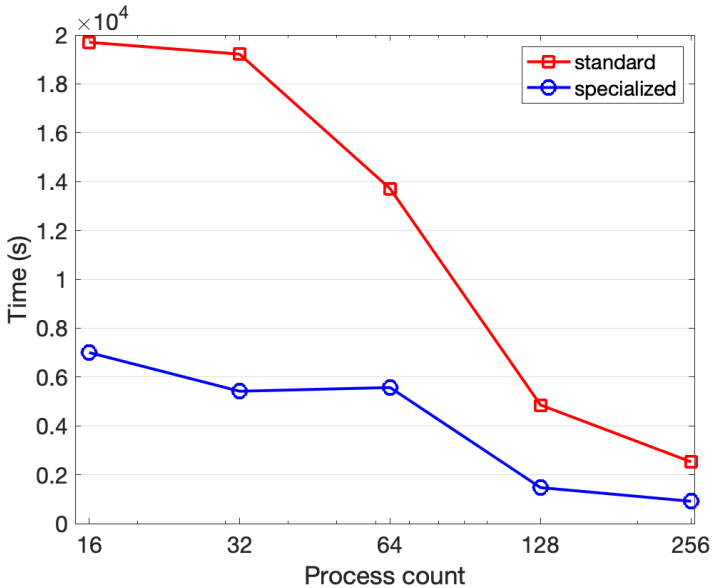


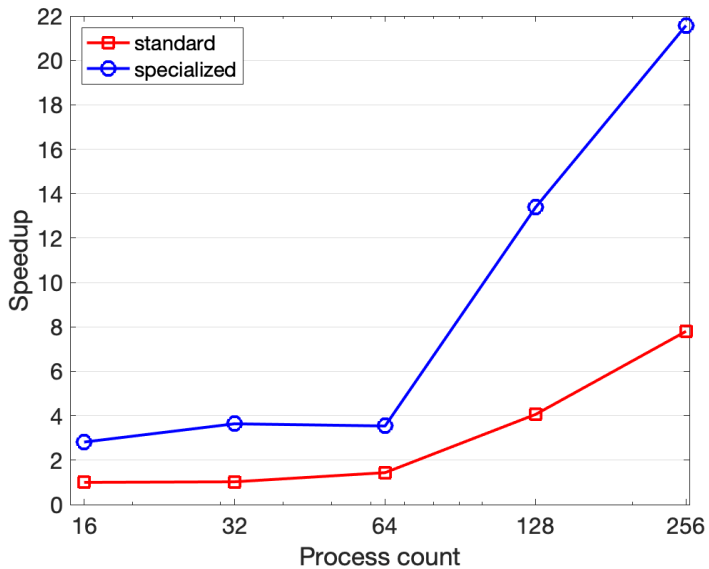
$n = 240,369$, $n_e = 700$
computing 100 eigenvalues at a time with $m = 200$ and $s = 5$

IV. Lanczos algorithm with EED and MPK

7. Preliminary results on strong-parallel scaling of
   sTRLED $=$ s-step-TRLan $+$ EED $+$ CA-MPK

   speedup over standard MPK `2D Laplacian`:





$n = 250,000$, $n_e = 5,000$,
computing 500 eigenvalues at a time with $m = 1000$ and $s = 5$

V. Concluding remarks

V. Concluding remarks

1. Many eigenpairs computation in emerging applications, a challenging problem even when vast computational resources are available.

2. Two techniques discussed in this talk:

   - Explicit external deflation (EED) for reliably moving away computed eigenpairs,
   - a communication-avoiding matrix powers kernel (CA-MPK) for fast sparse-plus-low-rank MPK

3. *The capability of being able to efficiently compute large number of eigenvalues will not just be appealing, but also mandatory for the next generation of eigensolvers.*

VI. References

VI. References

1. B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM Classics Edition, 1998.
2. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Revised edition, SIAM, 2011.
3. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst, eds. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, 2000.
4. C. Leiserson, S. Rao, S. Toledo, Efficient out-of-core algorithms for linear relaxation using blocking covers, J. Comput. Syst. Sci. Int. Vol.54, pp.332-–344, 1997.
5. N. Knight, E. Carson, J. Demmel, Exploiting data sparsity in parallel matrix powers computations, PPAM, pp.15–25, 2014.

VI. References

6. S. Kim, A. T. Chronopoulos, A class of Lanczos-like algorithms implemented on parallel computers, Parallel Computing, Vol.17, pp.763–778, 1991.
7. I. Yamazaki, K. Wu, A communication-avoiding thick-restart Lanczos method on a distributed-memory system, in: Workshop on Algorithms and Programming Tools for next-generation high-performance scientific and software (HPCC), pp.345–354, 2011.
8. E. Carson and J. W. Demmel, Accuracy of the s-step Lanczos method for the symmetric eigenproblem in finite precision, SIAM J. Matrix Anal. Appl. Vol.36, pp.783-819, 2015.
9. Z. Bai, J. Dongarra, D. Lu and I. Yamazaki, Matrix powers kernels for thick-restart Lanczos with explicit external deflation, IPDPS, May 2019
10. C. Lin, D. Lu and Z. Bai, Backward stability analysis of explicit external deflation for symmetric eigenvalue problems, *to appear soon*